Électronique numérique

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

Machine virtuelle GNU/Linux à
`http://jmfriedt.sequanux.org/stretch.ova`
(VirtualBox 5.0.24)

15 septembre 2016

Plan des interventions :

6 cours/TD :

1. Électronique numérique et conception du circuit
   aspects analogiques, consommation électrique, lecture de datasheet
   Survol des divers périphériques qui seront abordés (RS232, SPI, timer, ADC)
   représentation des données (tailles/encodage), masques, architecture
2. Rappel des bases du C :
   Présentation du compilateur gcc, compilation séparée, étapes de compilation
   Fonctionnement du compilateur (optimisations, passages de paramètres)

---

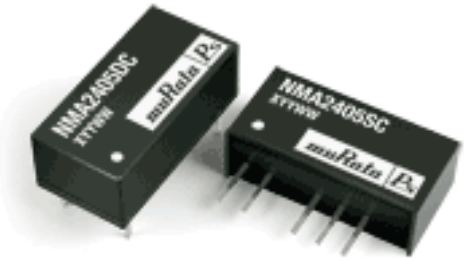
3. bus de communication & bibliothèques
   protocoles de communication série
   libopencm3, newlib & stubs, ressources requises par les bibliothèques
4. méthodes de développement séparant algorithmique et bas niveau
   timer, gestion des horloges du STM32, ADC
   portabilité et test du code sur PC, arithmétique sur systèmes embarqués
5.
6.

4 TP :

1. commandes de base sous unix & compilateur C
2. prise en main du microcontrôleur STM32 : horloges, UART, SPI
3. timer, conversion analogique numérique et exploitation des données
4. caractérisation du microsystème

Retour sur l'alimentation DC-DC

NMA0512SC



Isolated 1W Dual Output DC/DC Converters

Recommended | In Production | ?

DATASHEET (PDF) | ADD TO COMPARE LIST

CONTACT US | SAMPLE REQUEST

TECHNICAL SUPPORT | REQUEST A QUOTE

STOCKCHECK: WHERE TO BUY | ROHS-CERTIFICATE

DC-DC APPLICATION NOTES | 3D MODELS

Technical Information

Input Voltage Range (V)	4.5 to 5.5
Design Origin	Murata Power Solutions
Dimensions (inch)	0.24 x 0.77 x 0.39
Dimensions (mm)	6 x 19.5 x 10
Efficiency (%)	77

Efficacité : 77%

Retour sur l'alimentation DC-DC

MAX5035

1A, 76V, High-Efficiency MAXPower Step-Down DC-DC Converter

Data Sheet Subscribe Active: In Production. But some versions of the family are Not Recommended for New Designs.

OVERVIEW | KEY SPECS | DESIGN RESOURCES | ORDER

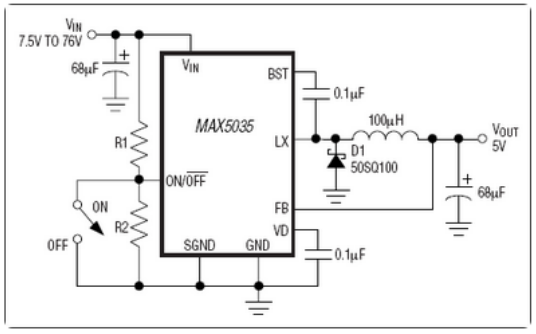
Description

The MAX5035 easy-to-use, high-efficiency, high-voltage, step-down DC-DC converter operates from an input voltage up to 76V and consumes only 270µA quiescent current at no load. This pulse-width modulated (PWM) converter operates at a fixed 125kHz switching frequency at heavy loads, and automatically switches to pulse-skipping mode to provide low quiescent current and high efficiency at light loads. The MAX5035 includes internal frequency compensation simplifying circuit implementation. The device uses an internal low-on-resistance, high-voltage, DMOS transistor to obtain high efficiency and reduce overall system cost. This device includes undervoltage lockout, cycle-by-cycle current limit, hiccup mode output short-circuit protection, and thermal shutdown.

The MAX5035 delivers up to 1A output current. The output current may be limited by the maximum power dissipation capability of the package. External shutdown is included, featuring 10µA (typ) shutdown current. The MAX5035A/B/C versions have fixed output voltages of 3.3V, 5V, and 12V, respectively, while the MAX5035D/E versions have an adjustable output voltage from 1.25V to 13.2V.

The MAX5035 is available in space-saving 8-pin SO and 8-pin plastic DIP packages and operates over the automotive (-40°C to +125°C) temperature range.

MAX5035: Typical Operating Circuit



Enlarge+

Key Features

- Wide 7.5V to 76V Input Voltage Range
- Fixed (3.3V, 5V, 12V) and Adjustable (1.25V to 13.2V) Versions
- 1A Output Current
- Efficiency Up to 94%

Applications/Uses

- Automotive
- Consumer Electronics
- Distributed Power
- Industrial

Efficacité : 94%

Code ASCII

Comment afficher la fréquence à l'écran

Table ASCII : correspondance entre un nombre $\in [0..127]$ et un symbole

```
ASCII(7)         Linux Programmer's Manual         ASCII(7)

NAME
ascii - the ASCII character set encoded in octal, decimal,
and hexadecimal

     2 3 4 5 6 7        30 40 50 60 70 80 90 100 110 120
    -------------      --------------------------------
  0:   0 @ P ` p     0:    (  2  <  F  P  Z  d   n   x
  1: ! 1 A Q a q     1:    )  3  =  G  Q  [  e   o   y
  2: " 2 B R b r     2:    *  4  >  H  R  \  f   p   z
  3: # 3 C S c s     3: !  +  5  ?  I  S  ]  g   q   {
  4: $ 4 D T d t     4: "  ,  6  @  J  T  ^  h   r   |
  5: % 5 E U e u     5: #  -  7  A  K  U  _  i   s   }
  6: & 6 F V f v     6: $  .  8  B  L  V  `  j   t   ~
  7:   7 G W g w     7: %  /  9  C  M  W  a  k   u  DEL
  8: ( 8 H X h x     8: &  0  :  D  N  X  b  l   v
  9: ) 9 I Y i y     9: '  1  ;  E  O  Y  c  m   w
  A: * : J Z j z
  B: + ; K [ k {
  C: , < L \ l |
  D: - = M ] m }
  E: . > N ^ n ~
  F: / ? O _ o DEL
```

- Découper chaque quartet en symboles individuels ⇒ hexadécimal
  Exercice
- Effectuer les divisions successives ⇒ décimal
  Exercice

Un microcontrôleur ne sait pas ce qu'est `printf()` (vers quel périphérique écrire ?) ⇒ utiliser une fonction `putchar`.

Code ASCII

Comment afficher la fréquence à l'écran

Table ASCII : correspondance entre un nombre $\in [0..127]$ et un symbole

```
ASCII(7)         Linux Programmer's Manual         ASCII(7)

NAME
ascii - the ASCII character set encoded in octal, decimal,
and hexadecimal

     2 3 4 5 6 7        30 40 50 60 70 80 90 100 110 120
   -------------      --------------------------------
  0:   0 @ P ` p     0:    (  2  <  F  P  Z  d   n   x
  1: ! 1 A Q a q     1:    )  3  =  G  Q  [  e   o   y
  2: " 2 B R b r     2:    *  4  >  H  R  \  f   p   z
  3: # 3 C S c s     3: !  +  5  ?  I  S  ]  g   q   {
  4: $ 4 D T d t     4: "  ,  6  @  J  T  ^  h   r   |
  5: % 5 E U e u     5: #  -  7  A  K  U  _  i   s   }
  6: & 6 F V f v     6: $  .  8  B  L  V  `  j   t   ~
  7:   7 G W g w     7: %  /  9  C  M  W  a  k   u  DEL
  8: ( 8 H X h x     8: &  0  :  D  N  X  b  l   v
  9: ) 9 I Y i y     9: '  1  ;  E  O  Y  c  m   w
  A: * : J Z j z
  B: + ; K [ k {
  C: , < L \ l |
  D: - = M ] m }
  E: . > N ^ n ~
  F: / ? O _ o DEL
```

- Découper chaque quartet en symboles individuels ⇒ hexadécimal

```
void affiche(short s)
{int k;char c;
 for (k=3;k>=0;k--)
   {c=s>>(4*k)&0x0f;
    if (c<10) putchar(c+'0');
       else putchar(c+'A'-10);
   }
}
```

- Effectuer les divisions successives ⇒ décimal

Exercice

Code ASCII

Comment afficher la fréquence à l'écran

Table ASCII : correspondance entre un nombre $\in [0..127]$ et un symbole

```
ASCII(7)         Linux Programmer's Manual         ASCII(7)

NAME
ascii - the ASCII character set encoded in octal, decimal,
and hexadecimal

     2 3 4 5 6 7        30 40 50 60 70 80 90 100 110 120
   -------------      --------------------------------
  0:   0 @ P ` p     0:    (  2  <  F  P  Z  d   n   x
  1: ! 1 A Q a q     1:    )  3  =  G  Q  [  e   o   y
  2: " 2 B R b r     2:    *  4  >  H  R  \  f   p   z
  3: # 3 C S c s     3: !  +  5  ?  I  S  ]  g   q   {
  4: $ 4 D T d t     4: "  ,  6  @  J  T  ^  h   r   |
  5: % 5 E U e u     5: #  -  7  A  K  U  _  i   s   }
  6: & 6 F V f v     6: $  .  8  B  L  V  `  j   t   ~
  7:   7 G W g w     7: %  /  9  C  M  W  a  k   u  DEL
  8: ( 8 H X h x     8: &  0  :  D  N  X  b  l   v
  9: ) 9 I Y i y     9: '  1  ;  E  O  Y  c  m   w
  A: * : J Z j z
  B: + ; K [ k {
  C: , < L \ l |
  D: - = M ] m }
  E: . > N ^ n ~
  F: / ? O _ o DEL
```

- Découper chaque quartet en symboles individuels ⇒ hexadécimal

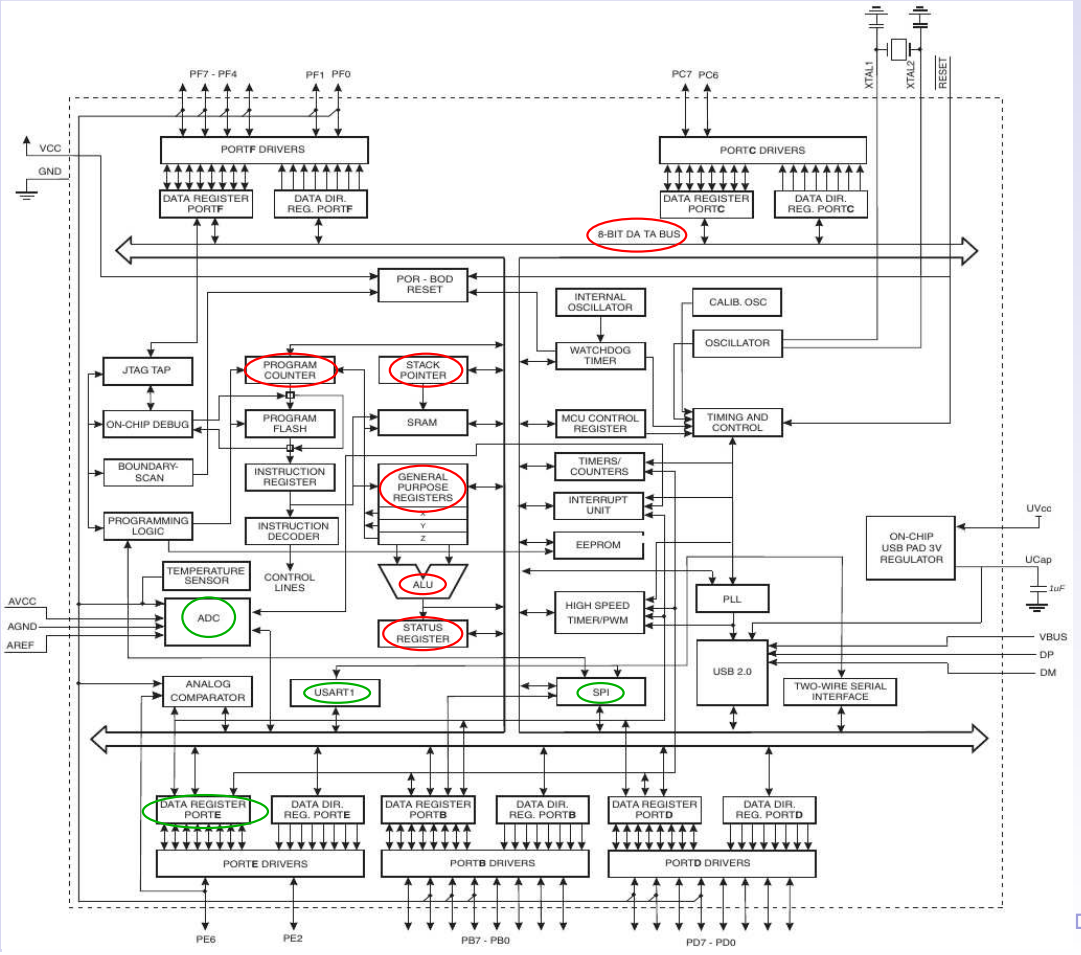
```
void affiche(unsigned short s)
{int k;char c;
 for (k=3;k>=0;k--)
   {c=s>>(4*k)&0x0f;
    if (c<10) putchar(c+'0');
       else putchar(c+'A'-10);
   }
}
```

- Effectuer les divisions successives ⇒ décimal

```
void affiche(unsigned short s)
{int k=10000;char c; // k<65536
 while (k>0)
   {c=s/k;putchar(c+'0');
    s=s-c*k;
    k/=10;
   }
}
```

Architecture interne d'un microcontrôleur

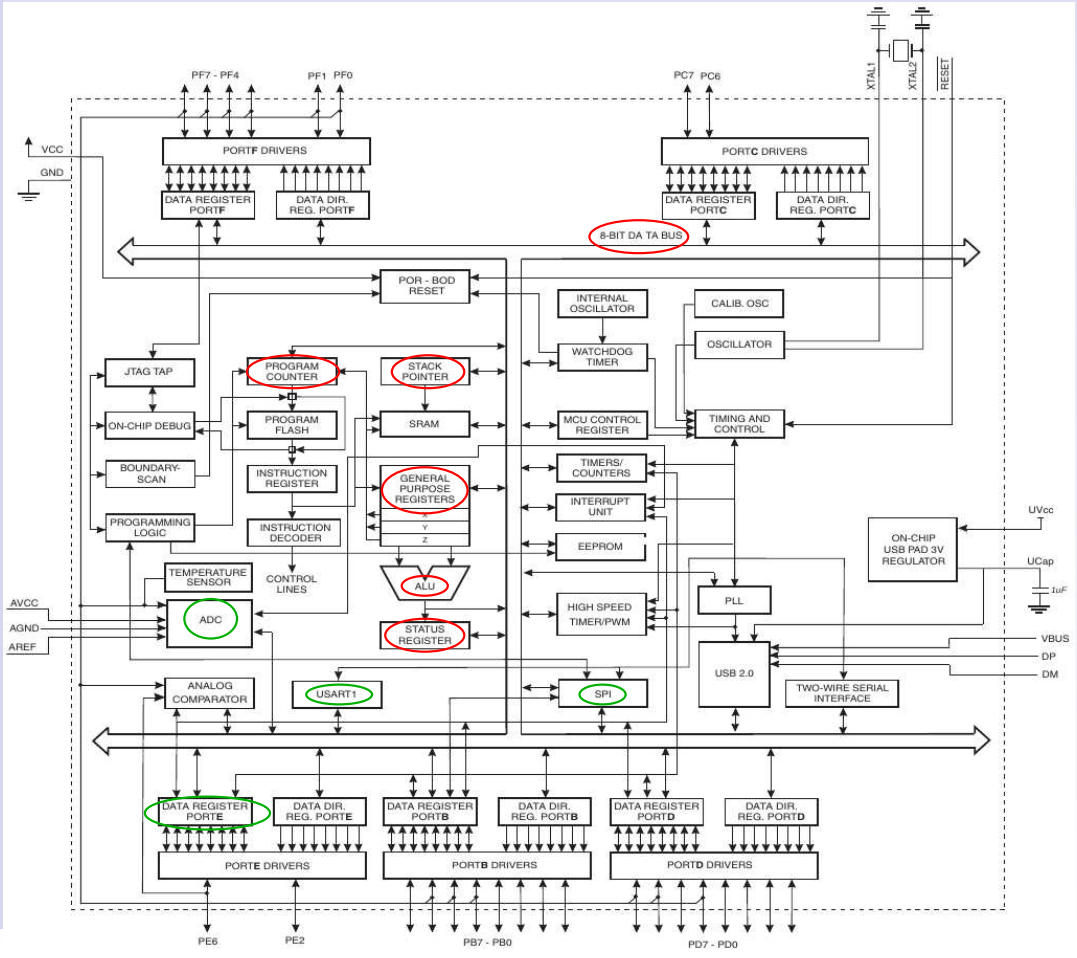
- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



- ALU : Unité Arithmérique et Logique
- GPIO
- timer
- ADC
- PC
- SP

Architecture interne d'un microcontrôleur

- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



6.4 GPIO registers

This section gives a detailed description of the GPIO registers.
For a summary of register bits, register address offsets and reset values, refer to *Table 27.*

The GPIO registers can be accessed by byte (8 bits), half-words (16 bits) or words (32 bits).

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..C and H)

Address offset: 0x00

Reset values:

- 0x0C00 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..C and H)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

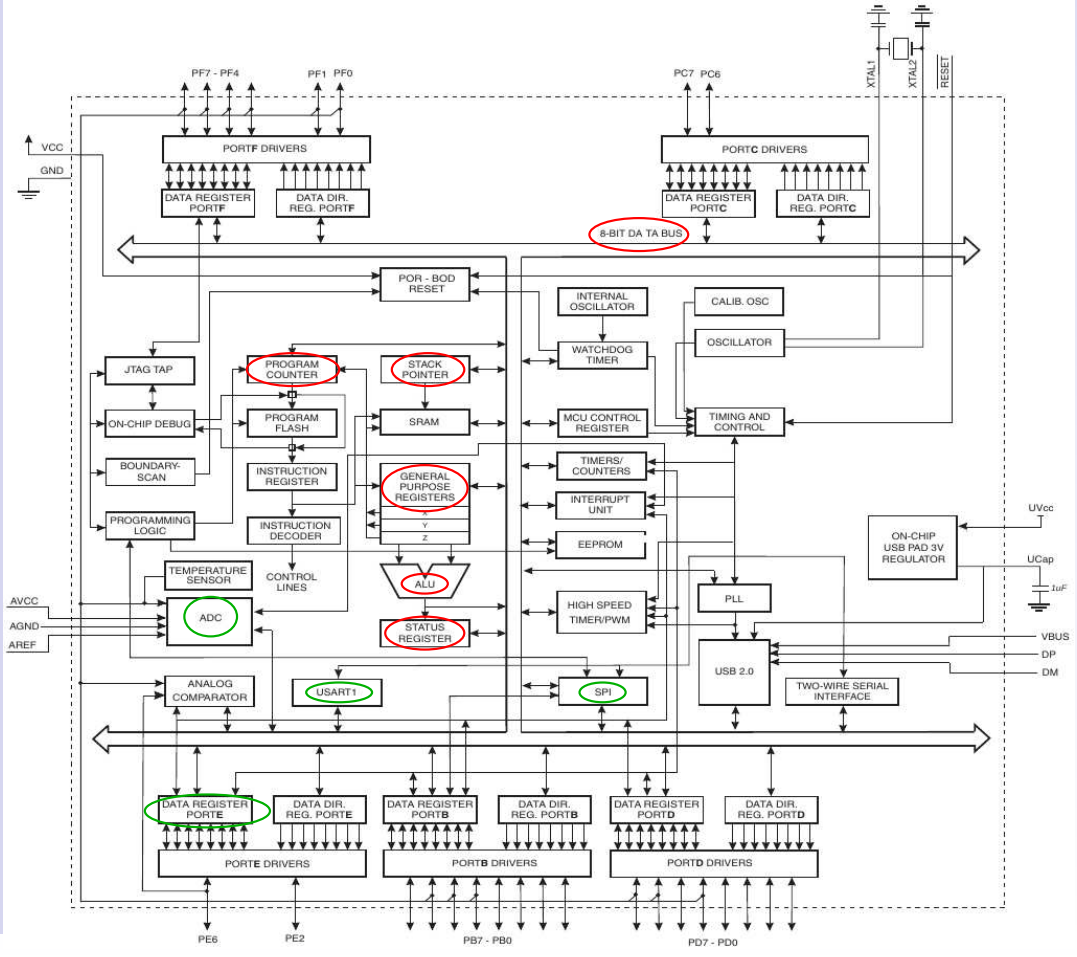
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy**: Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

RM0401 : datasheet STM32F410

Architecture interne d'un microcontrôleur

- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



Memory map and register boundary addresses

See the datasheet corresponding to your device for a comprehensive diagram of the memory map.

The following table gives the boundary addresses of the peripherals available in the devices.

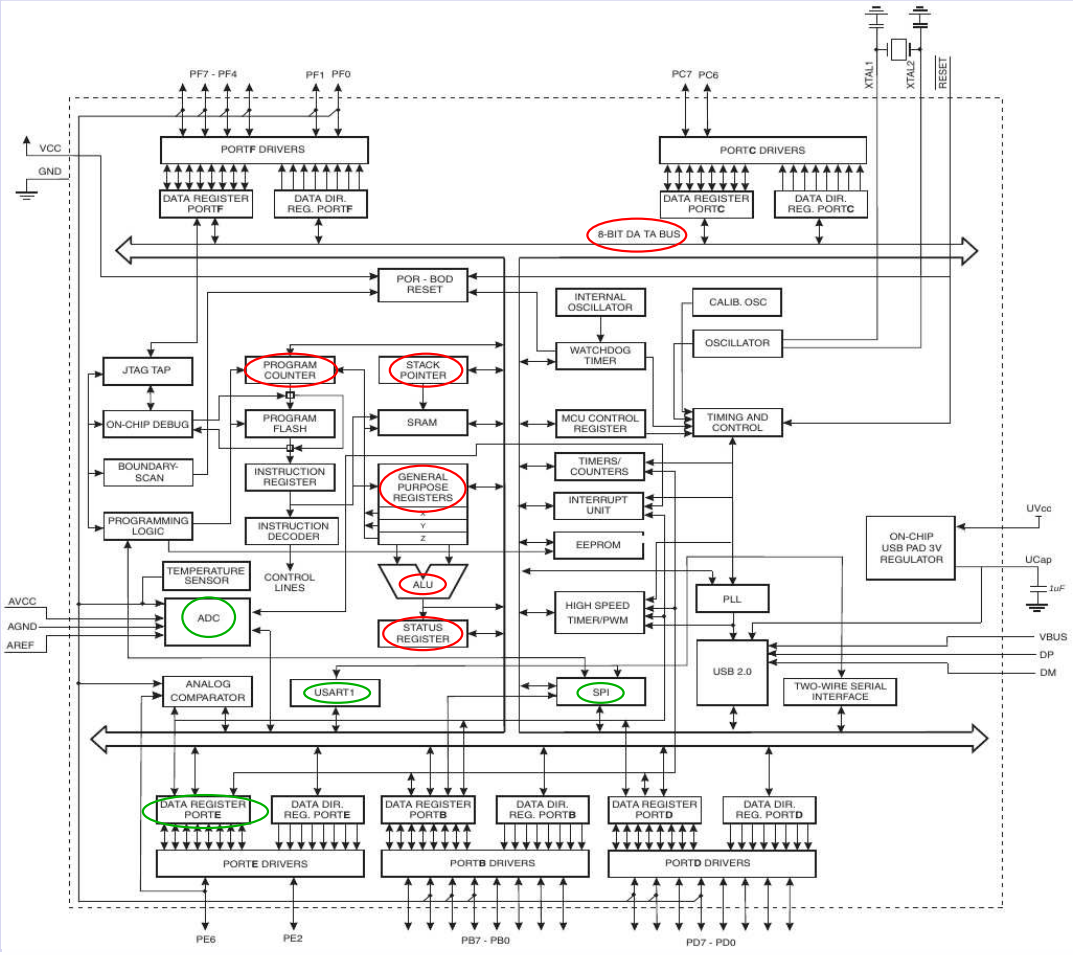
Table 1. Register boundary addresses

Bus	Boundary address	Peripheral
	0xE010 0000 - 0xFFFF FFFF	Reserved
Cortex®-M4	0xE000 0000 - 0xE00F FFFF	Cortex-M4 internal peripherals
	0x5000 0000 - 0xDFFF FFFF	Reserved
AHB1	0x4008 0400 - 0x4FFF FFFF	Reserved
	0x4008 0000 - 0x4008 03FF	RNG
	0x4002 6800 - 0x4007 FFFF	Reserved
	0x4002 6400 - 0x4002 67FF	DMA2
	0x4002 6000 - 0x4002 63FF	DMA1
	0x4002 5000 - 0x4002 4FFF	Reserved
	0x4002 3C00 - 0x4002 3FFF	Flash interface register
	0x4002 3800 - 0x4002 3BFF	RCC
	0x4002 3400 - 0x4002 37FF	Reserved
	0x4002 3000 - 0x4002 33FF	CRC
	0x4002 2800 - 0x4002 2FFF	Reserved
	0x4002 2400 - 0x4002 27FF	LPTIM1
	0x4002 2000 - 0x4002 23FF	Reserved
	0x4002 1C00 - 0x4002 1FFF	GPIOH
	0x4002 0C00 - 0x4002 1BFF	Reserved
	0x4002 0800 - 0x4002 0BFF	GPIOC
	0x4002 0400 - 0x4002 07FF	GPIOB
	0x4002 0000 - 0x4002 03FF	GPIOA

RM0401 : datasheet STM32F410

Architecture interne d'un microcontrôleur

- Lien entre le matériel et le logiciel : bus d'adresse, de données et de contrôle (où, quoi, comment)
- Bus = ensemble de fils reliant deux périphériques (liaison parallèle)



/libopencm3/stm32/common/gpio_common_f234.h : #define GPIOC GPIO_PORT_C_BASE

/libopencm3/stm32/f4/memorymap.h : #define GPIO_PORT_C_BASE (PERIPH_BASE_AHB1 + 0x0800)

/libopencm3/stm32/f4/memorymap.h : #define PERIPH_BASE_AHB1 (PERIPH_BASE + 0x20000)

/libopencm3/stm32/f4/memorymap.h : #define PERIPH_BASE (0x40000000U)

$\Rightarrow$ GPIOC = 0x40020800

Comment connaître l'adresse d'un périphérique ...

Liste des registres dans la datasheet du microcontrôleur (Atmega32U4)

$\Rightarrow$ utilisation des masques pour définir un bit individuel

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x16 (0x36)	TIFR1	-	-	ICF1	-	OCF1C	OCF1B	OCF1A	TOV1
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-
0x13 (0x33)	Reserved	-	-	-	-	-	-	-	-
0x12 (0x32)	Reserved	-	-	-	-	-	-	-	-
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	-	-	PORTF1	PORTF0
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	-	-	DDF1	DDF0
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	-	-	PINF1	PINF0
0x0E (0x2E)	PORTE	-	PORTE6	-	-	-	PORTE2	-	-
0x0D (0x2D)	DDRE	-	DDE6	-	-	-	DDE2	-	-
0x0C (0x2C)	PINE	-	PINE6	-	-	-	PINE2	-	-
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x08 (0x28)	PORTC	PORTC7	PORTC6	-	-	-	-	-	-
0x07 (0x27)	DDRC	DDC7	DDC6	-	-	-	-	-	-
0x06 (0x26)	PINC	PINC7	PINC6	-	-	-	-	-	-
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x02 (0x22)	Reserved	-	-	-	-	-	-	-	-
0x01 (0x21)	Reserved	-	-	-	-	-	-	-	-
0x00 (0x20)	Reserved	-	-	-	-	-	-	-	-

Extrait de datasheet Atmega 32U4

Électronique numérique : gcc

- Ne pas s'handicaper avec un compilateur propriétaire compatible avec une unique plateforme matérielle : gcc [1]
- Maîtriser une large gamme de processeurs pour répondre à tous les besoins
  - périphériques dédiés autour d'un cœur de processeur générique (8051, ARM)
  - cœur répondant aux besoins de puissance de calcul
  - vitesse la plus faible possible pour réduire la consommation

aarch64	fr30	mcore	nvptx	stormy16
alpha	frv	mep	pa	tilegx
arc	**h8300**	microblaze	pdp11	tilepro
arm	**i386**	**mips**	rl78	v850
avr	**ia64**	mmix	rs6000	vax
bfin	iq2000	mn10300	rx	visium
c6x	lm32	moxie	s390	xtensa
cr16	m32c	**msp430**	sh	
cris	m32r	nds32	**sparc**	
epiphany	**m68k**	nios2	spu	

1. https://gcc.gnu.org/backends.html

Outils de développement

Sans système d'exploitation (*bare-metal*)

- `gcc+binutils+bibliothèques (newlib)`
  Capacité à recompiler sa chaîne de compilation sur toute plateforme[2]/pour toute plateforme supportée :
  `./configure --prefix=/usr/local/arm --target=arm-elf`
- un programmeur pour communiquer avec le bootloader (`avrdude`, `st-util`, OpenOCD ...), simple à réimplémenter si nécessaire
- outils périphériques (`gdb`)

Langage C fournit le meilleur compromis entre abstraction (variables, boucles, branchements conditionnels) et accès au matériel (pointeurs).

Quelques subtilités sur l'embarqué : `volatile`, `*(type*)addr=valeur;`

MAIS il faut tout comprendre (pas de pilote, bibliothèques à adapter au matériel)

2. automatisé pour ARM M3/M4 par `summon-arm-toolchain` à `https://github.com/jmfriedt/summon-arm-toolchain`